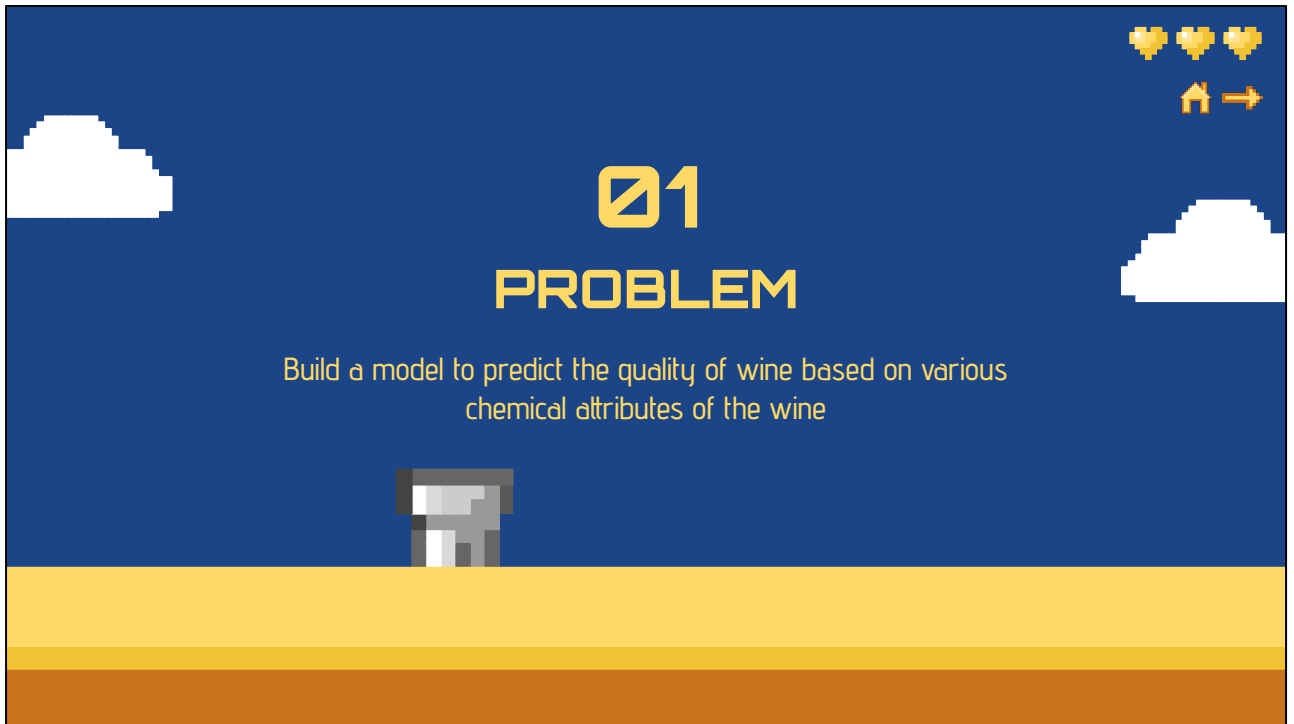





Wine Quality Classification

CSC44700 - Final Project

Alice Liu



Winemaking is considered an art for winemakers. Each bottle has different tastes. This sensory experience lies in the realm of science. The goal of this project is to unravel the relationship between wine chemistry and perceived quality of the wine. My project focuses on building a model to predict the quality of wine based on various chemical attributes of the wine.



DATASET

Original Dataset

- Shape: (1143, 13)
- Columns: 13
 - 'fixed acidity', 'volatile acidity', 'citric acidity', 'residual sugar', 'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density', 'pH', 'sulphates', 'alcohol', 'ld'
 - Target: 'quality'
- No NULL

My first step when approaching this project, is to analyze the dataset; do some initial exploratory analysis.

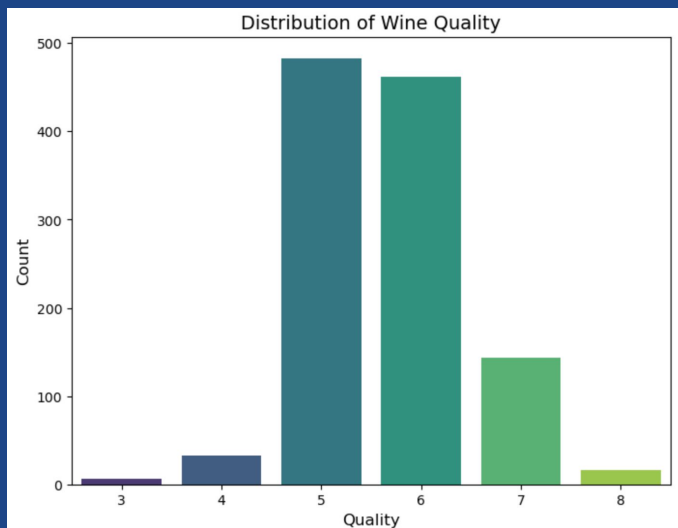
This includes:

1. Analyzing the shape of the dataset
2. The different columns
3. What each column means in wine and during the winemaking process
4. What is the target column--which I identified to be column `quality`.
5. The number of NULL values per column
6. The statistical description of each column--meaning (standard deviation, mean, things of that sort)

During my EDA process, I've discovered that there was:

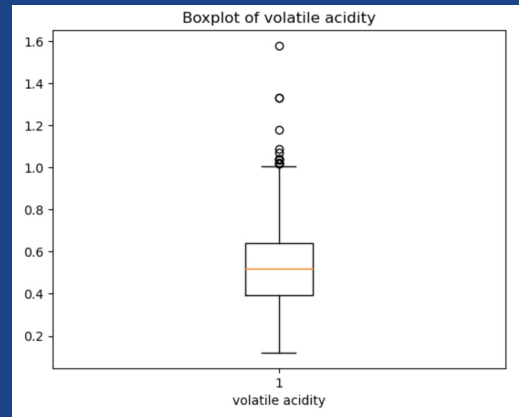
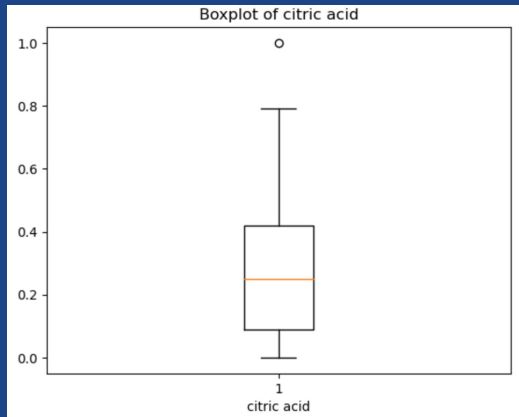
- No NULL values
- The column Id was useless, and so I dropped the column.

Dataset Distribution of Unique Value from 'quality'



This showcases the distribution of unique values from our target variable, 'quality'. I did this to understand what were the different values that helped indicate good/bad quality.

Initial EDA and Cleaning



Furthermore, I wanted to analyze the existence of outliers in the dataset for each column. The reasoning behind this is because, outliers can significantly impact the performance, accuracy, and reliability of the model.

Now up there/here, are box plot images for 2 out of 12 columns. These are examples of what I had in my dataset.

Initial EDA and Cleaning

```
# DROP Outliers for all columns using IQR
def drop_outliers(df):
    df_new = pd.DataFrame()

    for col in df.columns:
        q1 = df[col].quantile(0.25)
        q3 = df[col].quantile(0.75)
        IQR = q3 - q1
        lower_bound = q1 - 1 * IQR # 1.5 and 2 didn't work; 1 worked the best for threshold
        upper_bound = q3 + 1 * IQR # 1.5 and 2 didn't work; 1 worked the best for threshold

        # drop outliers
        df_new[col] = df[(df[col] >= lower_bound) & (df[col] <= upper_bound)][col]

    return df_new

df_clean = drop_outliers(df)
df_clean = df_clean.dropna()
plt_boxplots(df_clean)
```

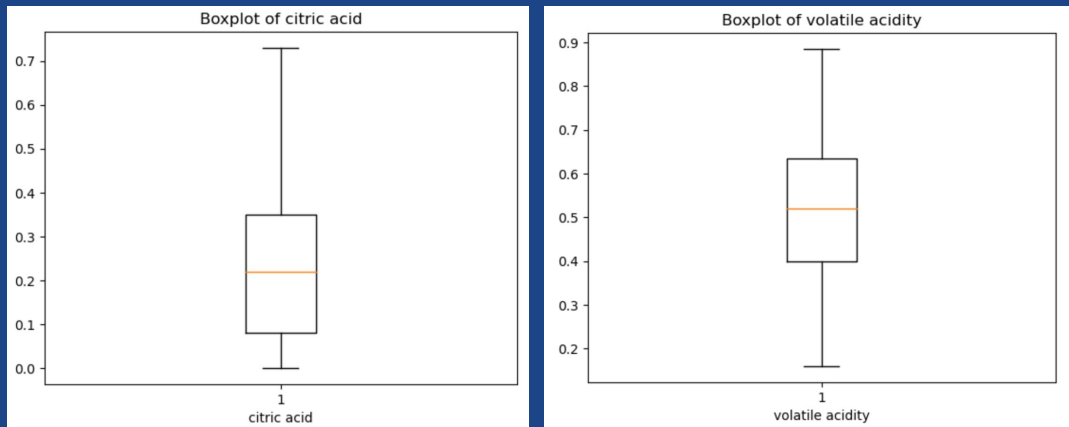
Now to get rid of the outliers in my dataset, I filtered out the outliers in my columns using IQR using my `drop_outliers` function. By using the IQR, it helps as an outlier detection by focusing on the middle 50% of the data distribution--so between the 1st and 3rd quartile.

For each column, I did the following:

1. Calculate the 1st and 3rd quartiles
2. Calculate the IQR using the 1st and 3rd quartiles
3. Set the lower and upper bounds for outlier detection using threshold of 1
 - a. The outliers are values above the upper bound and below the lower bound
4. I filtered the data frame to retain only the values within the upper and lower bounds

- IQR = interquartile range
-

Initial EDA and Cleaning



So, I put my dataframe through the function and created a new dataframe that holds the filtered outliers. As a result, my box plots--or the look of existing outliers--look a lot better. And for reference, these 2 box plot images are the same columns as I showed you in the previous slide, but filtered.



DATASET



Cleaned Dataset

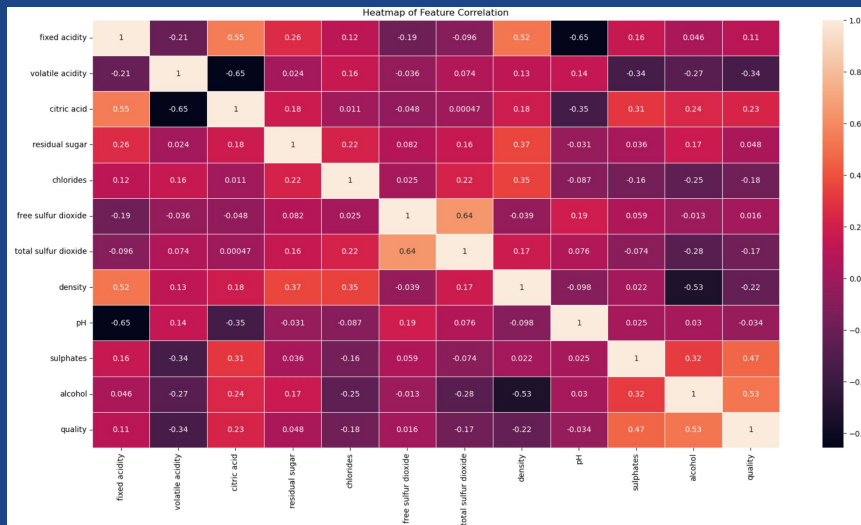
- Shape: (633, 12)
- Columns: 12
 - 'fixed acidity', 'volatile acidity', 'citric acidity', 'residual sugar', 'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density', 'pH', 'sulphates', 'alcohol'
 - Target: 'quality'



After all the initial exploratory data analysis and cleaning, my dataset now looks like this.



More EDA



+ Correlation

- Alcohol vs Quality
- Sulphates vs Quality
- Citric Acid vs Fixed Acidity
- Density vs Fixed Acidity
- Free SO2 vs Total SO2

- Correlation

- Fixed Acidity vs pH
- Citric Acid vs Volatile Acidity
- Alcohol vs Density

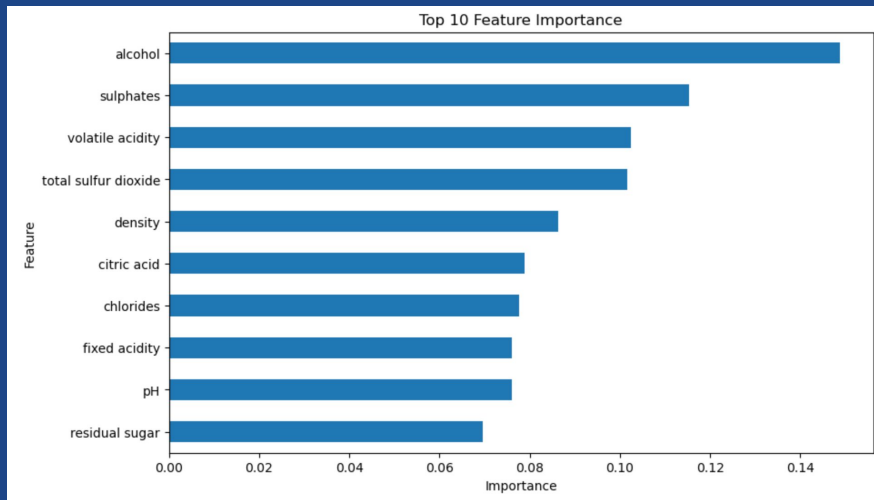
But of course, it's not over yet. I wanted to analyze the relationship and distribution of each columns, to gain a better understanding of what will be useful during modeling.

This includes a heatmap, for me to see the correlation of each variables. I took notice of what features had strong positive correlations, features with positive correlations with the target variable, and features with strong negative correlations. Although it's not shown in this PPT, if you want to take a look in my notebook, I also created scatter plots for each of these relationships to gain a better understanding of how these relationships worked in relevance to other features, such as pH.

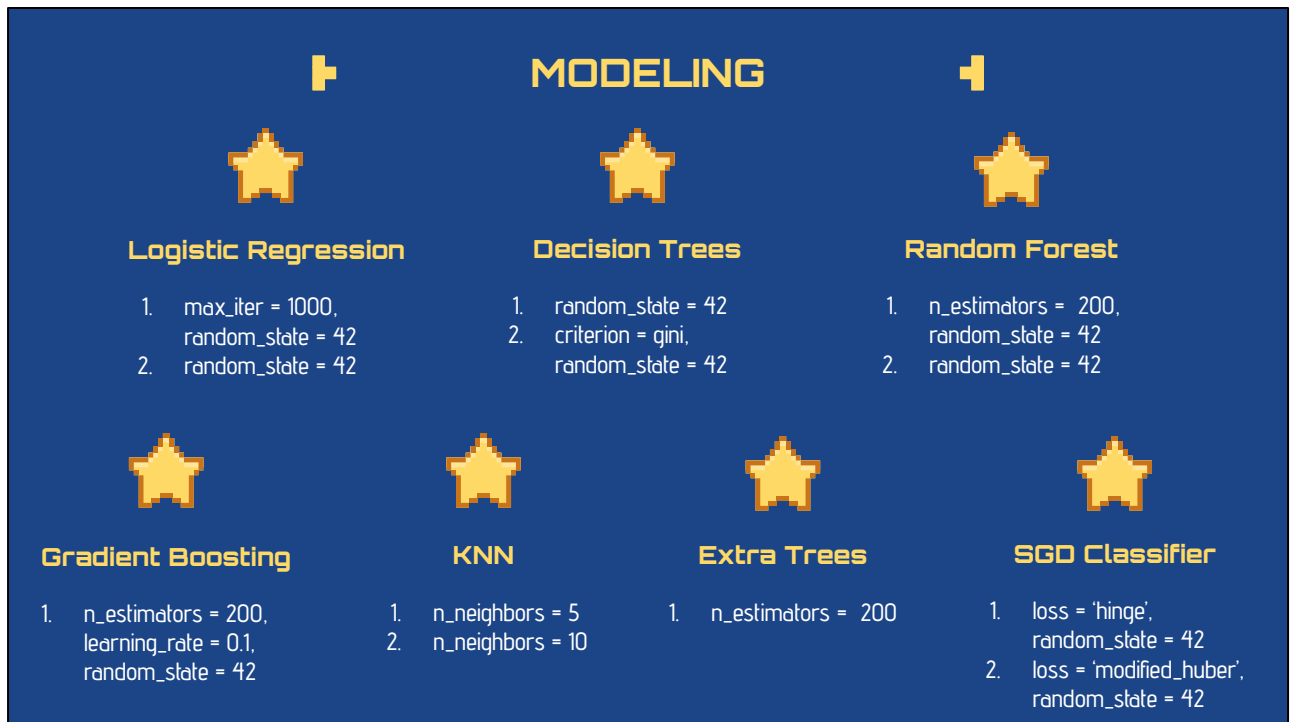
This shed some insight into feature selection in my modeling process.



FEATURE IMPORTANCE



I also created a feature importance bar chart using random forest. This generated the top 10 features that had importance to the target column. This shed some insight into feature selection in my modeling process.



For this project, I chose to test various models, some models had different parameters. The models I've decided to test are the following:

- Logistic regression, Decision trees, Random forest, Gradient Boosting, KNN, Extra Trees, and SGD Classifier

Extra Trees (aka. Extremely randomized tree):

- Type of ensemble learning method based on decision trees
- Similar to RF but introduces additional randomness during the tree-building process, leading to potentially higher diversity and improved performance in certain scenarios
- Random feature selection:
 - Unlike RF, which selects the best feature for splitting at each node, ET randomly selects feature subsets for splitting nodes. This introduces additional randomness, making the trees more diverse
- Random split points:
 - In addition to random feature selection, ET randomly chooses split points for each feature, regardless of the actual optimal split points. This further increases diversity in the trees
- Benefits:
 - By creating more diverse trees, ET is less prone to overfitting
 - Can lead to improved generalization

MODELING - Feature Selection

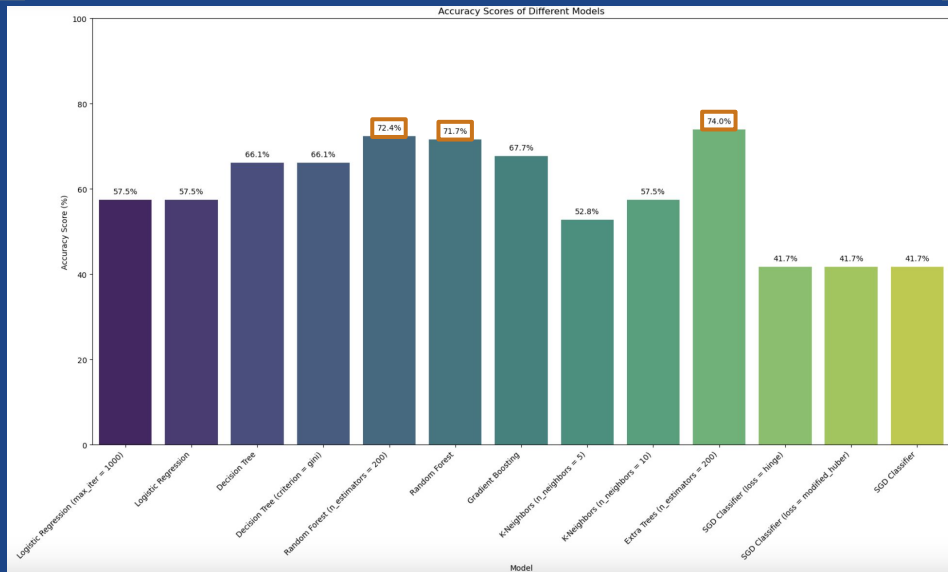
```
In [67]: # using TOP 3 FEATURES FROM FEATURE IMPORTANCE <- 72% in RF, 74% in ExtraTrees
X_clean = df_clean[['alcohol', 'sulphates', 'volatile acidity']]
y_clean = df_clean['quality']

X_train_clean, X_test_clean, y_train_clean, y_test_clean = train_test_split(X_clean, y_clean,
                                                                              test_size = 0.2, random_state = 42)
```

Now during my Feature Selection stage, I did a lot of experimenting. Meaning, I selected features that had strong positive correlation with the target variable, strong negative correlation, features that had strong positive correlation with the features that had strong positive correlation with the target variable, top 10, 9, 8, 5, 4, 3, 2, 1.... Features in the feature importance list, and I mix-and-matched.

The features that yielded the best result is on the screen. And those were the top 3 features from the feature importance list.

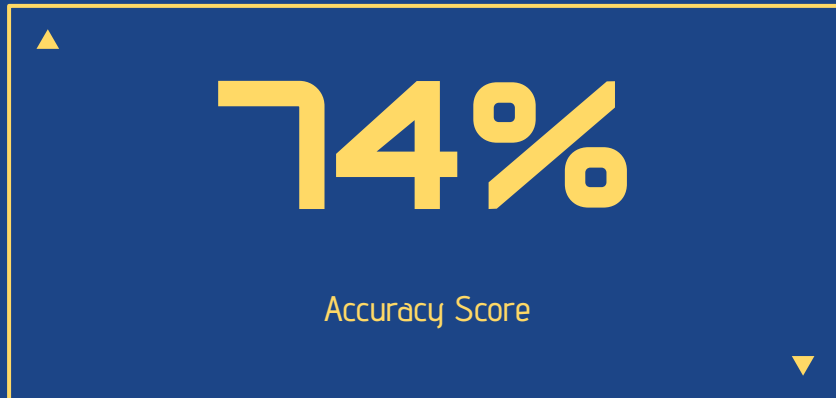
EVALUATION



Those features yielded these result for each model. A bit hard to see, but the top 3 performers was:

1. Extra Trees
2. Random Forest
3. Random Forest (n_estimators = 200)

HIGHEST: EXTRA TREES CLASSIFIER



And so, the highest accuracy score I was able to obtain is with the Extra Trees Classifier, which yielded a 74% accuracy score.

└ POINTS OF IMPROVEMENTS ─



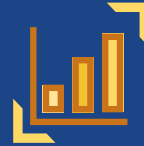
Feature Engineering

Explore additional feature engineering techniques (e.g., scaling, more effective handling of outliers, etc.)



Hyperparameter Tuning

Adjust the parameters, experiment using GridSearchCV



Model Selection

Try different models to see if they offer better performance, consider advanced ensemble techniques (e.g., stacking, blending multiple models)

Now of course, the value of 74% can improve. If there was more time:

- I would explore additional feature engineering techniques such as scaling
- Perform hyperparameter tuning using GridSearchCV
- And lastly, to experiment with different models to see if they offer better performance. Consider more advanced ensemble techniques such as stacking, since it seems that ExtraTrees--an ensemble method--yielded the best result.

Stacking:

- Combines multiple base models (learners) with different algorithms and then using a meta-model to learn how to best combine the predictions of these base models



That is all I have for today, thank you for tuning into this presentation.